Climate model design and the Earth System Modeling Framework

V. Balaji
Princeton University/GFDL

NASA HPC Summer School 2004
Greenbelt MD
23 July 2004

Weather and climate

“Climate is what you expect, weather is what you get.”

Climate is the study of long-term time averages.

The climate, however, has variations on all time scales: monsoons, ice ages.

More recently, the chemical and radiative properties of the atmosphere have been abruptly altered, with consequences that need to be explored.

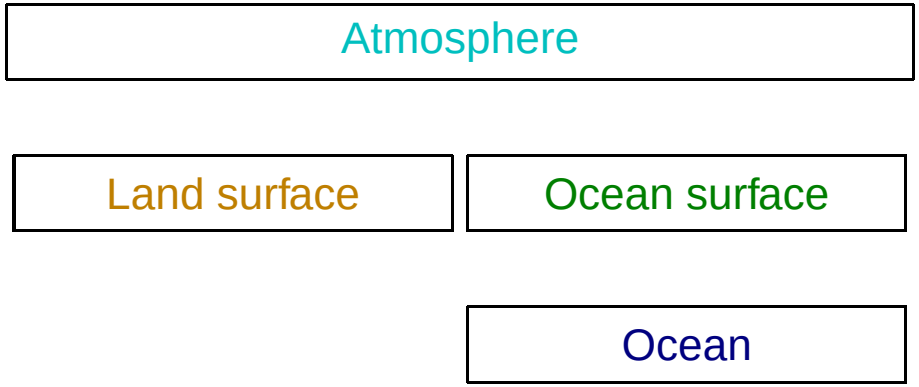
Components of the Earth system

Atmosphere atmospheric fluid dynamics and thermodynamics, moist processes, radiative transfer, transport and chemistry of trace constituents.

Ocean World ocean circulation, ocean biogeochemistry.

Land surface Surface processes, ecosystems, hydrology.

Ocean surface Sea ice, wave processes.



Complexity of climate simulations

Models have grown increasingly complex with time.

70s	80s	early 90s	late 90s	today	early 00s	late 00s
Atm	Atm	Atm	Atm	Atm	Atm	Atm
	Land	Land	Land	Land	Land	Land
		Ocn, Sealce	Ocn, Sealce	Ocn, Sealce	Ocn, Sealce	Ocn, Sealce
			Aerosols	Aerosols	Aerosols	Aerosols
					C Cycle	C Cycle
		Aerosols			Ecosystems	Ecosystems
		Land C			Chemistry	Chemistry
	Ocn, Sealce	Ocn Carbon			Ocn Eddies	Ocn Eddies
	Clouds	Chemistry	C Cycle	Ocn Eddies		Clouds

Components are developed “offline” (bottom left) and then are integrated into comprehensive coupled models.

Timescales and space scales

Aerosols seconds; μm – cm.

Clouds minutes to hours; 10 m – 100 km.

Ocean eddies hours to days; 10 km.

Weather patterns days; continent scales.

Ocean currents days; ocean basin scales.

Short-term climate seasonal-interannual; El Niño; monsoons.

Climate change Decadal-centennial; ocean overturning and deep water formation.

Paleoclimate Millennia to deep time; ice ages; orbital changes.

Discretization in space and time

Each process has its own intrinsic time and space scales.

Older models did not allow subcomponents to be on independent grids and timesteps.

- Old way: sharing of data through arrays in common blocks.

- New way: independent model grids connected by a coupler.

Each model *process* component becomes an independent *code* component that can be separately instantiated, initialized, stepped forward, and terminated.

Gridded and ungridded components

Components may be associated with a grid (gridded component), associated with more than one component (coupler), or none at all (local computations, e.g “physics”).

Each component has its own data dependencies, decomposition strategies and coupling requirements. It must be possible to take full advantage of these features in a parallel code.

“Physics”

Processes that are at space and timescales too small to resolve are treated in parameterized equations: gridscale average response to gridscale average inputs.

Examples:

- Turbulence
- Moist convection
- Radiative transfer
- Boundary layer

These generally can be written as local (pointwise) or have a vertical column orientation: no horizontal dependencies.

This approach has limits: when the physical process becomes non-local on the gridscale. This limitation is most keenly felt now with the treatment of convective clouds in the atmosphere, and turbulent eddies in the ocean. Cloud-resolving AGCMs and eddy-resolving OGCMs to study climate are still about a decade away.

Types of grids

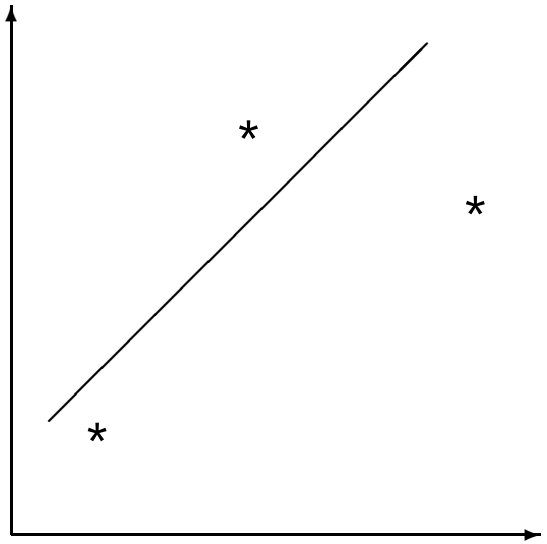
- Spectral grids.
- Lat-lon grids.
- Logically rectangular grids in generalized curvilinear coordinates:
  - Tripolar grids
  - Cubed sphere
- Reduced grids.
- Icosahedral and geodesic grids.
- Unstructured grids; catchment grids.

Data assimilation

Data can appear at unpredictable locations in time and space (radiosondes, buoys, satellites) and have no clear radius of influence ("location streams").

Models will "slosh" if incompatible with data.

Data assimilation involves bringing models and data into acquiescence. Assimilation algorithms can be treated as gridded components.



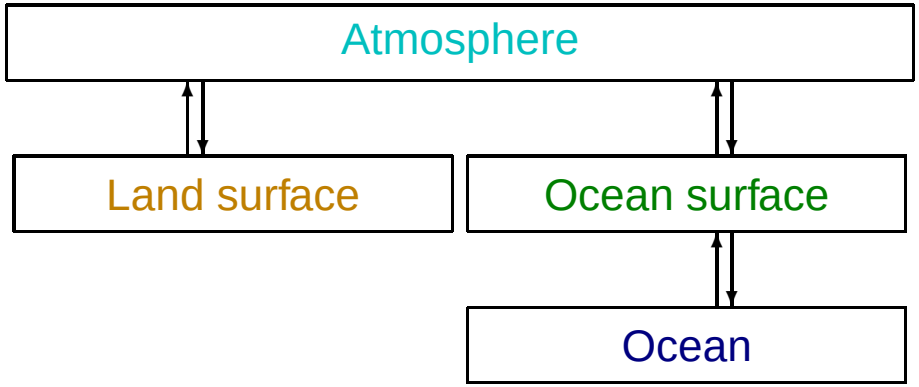
Model sizes and times

Climate Typically measured in simYears/wallDay. Resolution is adjusted till an experiment runs within a time useful for research. Current high-end research: $\sim 10^2$ variables integrated for $\sim 10^6$ timesteps on $\sim 10^6$ gridpoints. (Earth Simulator considerably raises this bar, but there is currently no relevant experience to compare against).

Weather Operational forecasting is measured in simHours/wallHour. Resolutions about 4×4 higher; $\sim 10^3$ timesteps.

Data exchange

Data exchanges between components take the form of fluxes of heat, momentum and tracers.



Conservation is very important for climate models; less so for weather.

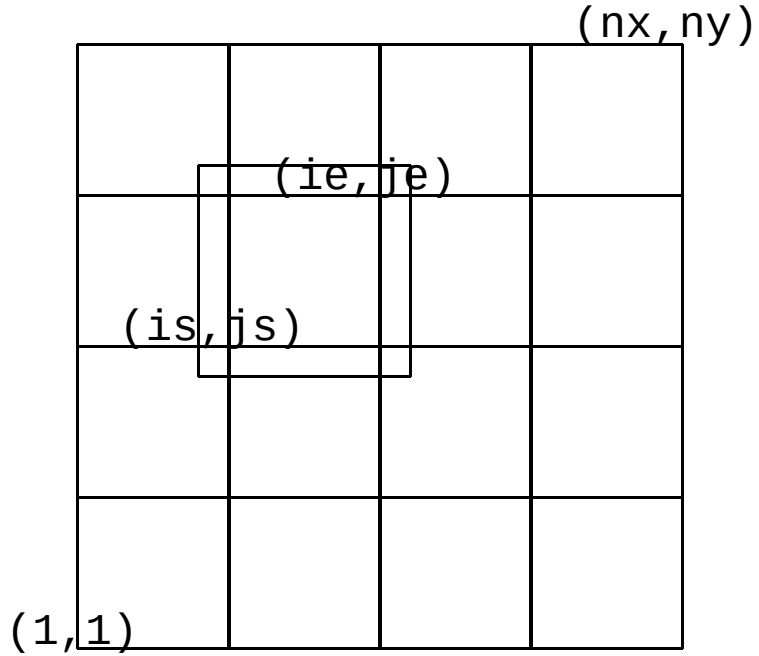
Memory models

Shared memory signal parallel and critical regions, private and shared variables. Canonical architecture: UMA, limited scalability.

Distributed memory domain decomposition, local caches of remote data ("halos"), copy data to/from remote memory ("message passing"). Canonical architecture: NUMA, scalable at cost of code complexity.

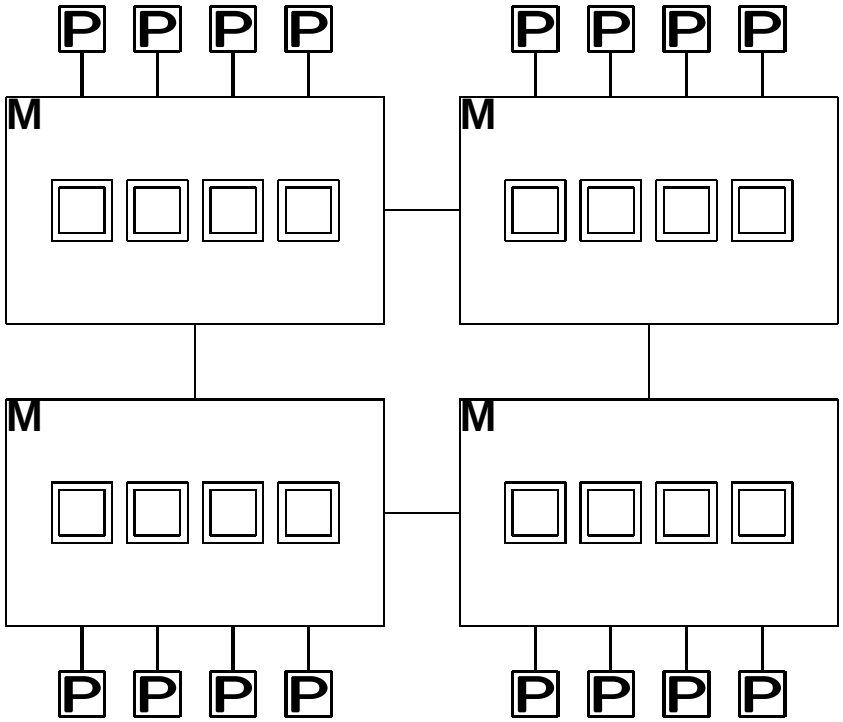
Distributed shared memory or ccNUMA message-passing, shared memory or remote memory access (RMA) semantics. Processor-to-memory distance varies across address space, must be taken into account in coding for performance. Canonical architecture: cluster of SMPs. Scalable at large cost in code complexity.

A 2D example



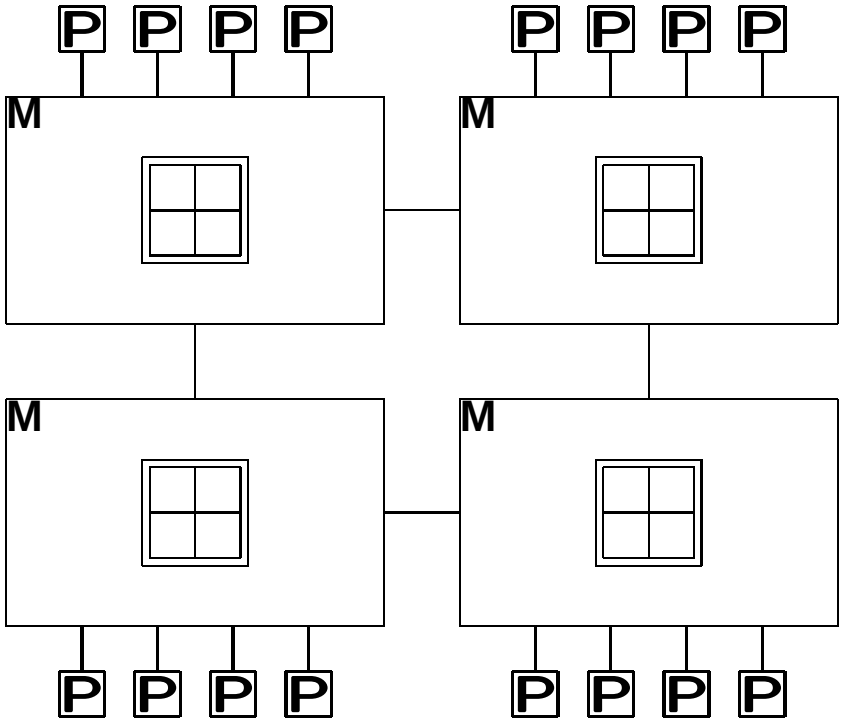
Consider a platform consisting of 16 PEs consisting of 4 mNodes of 4 PEs each. We also assume that the the entire 16-PE platform is a DSM or cc-NUMA aNode. We can then illustrate 3 ways to implement a `DistributedArray`. One PET is scheduled on each PE.

Distributed memory



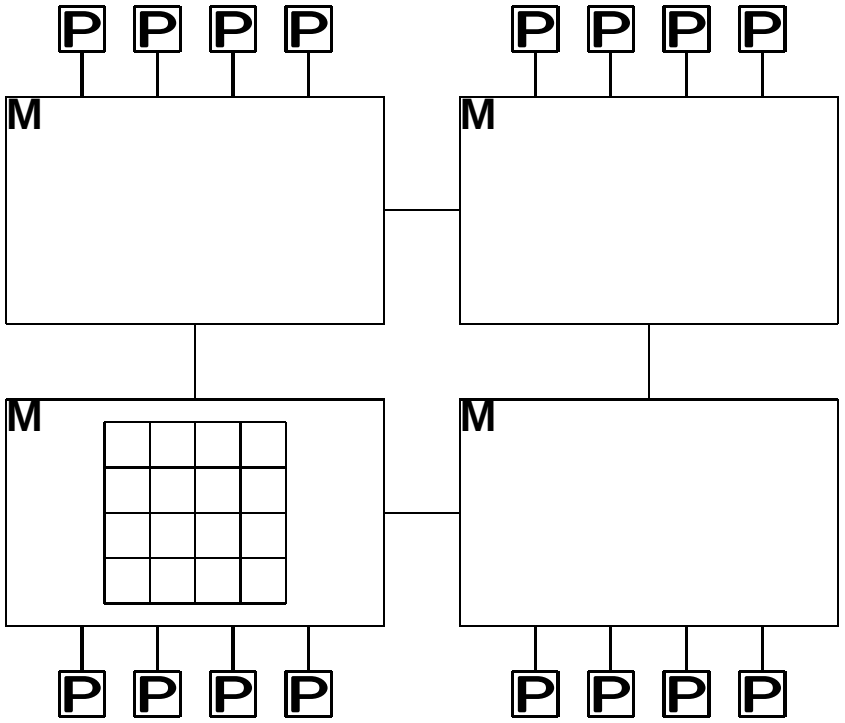
- each domain allocated as a separate array with halo, even within the same mNode.
- Performance issues: the message-passing call stack underlying MPI or another library may actually serialize when applied within an mNode.

Hybrid memory model



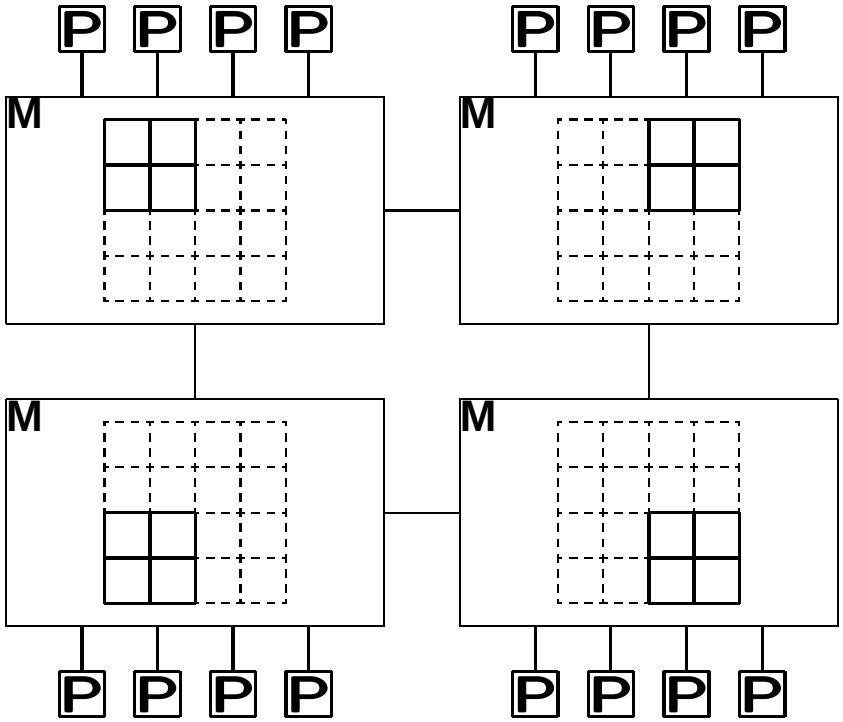
- shared across an mNode, distributed among mNodes.
- fewer and larger messages than distributed memory, may be less latency-bound.

Pure shared memory



Array is local to one mNode: other mNodes requires remote loads and stores. OK on platforms that are well-balanced in bandwidth and latency for local and remote accesses. ccNUMA ensures cache coherence across the aNode.

Intelligent memory allocation on DSM



Better memory locality: allocate each block of 4 domains on a separate page, and assign pages to different mNodes, based on processor-memory affinity.

Uniform interface to memory models: DEs and Layouts

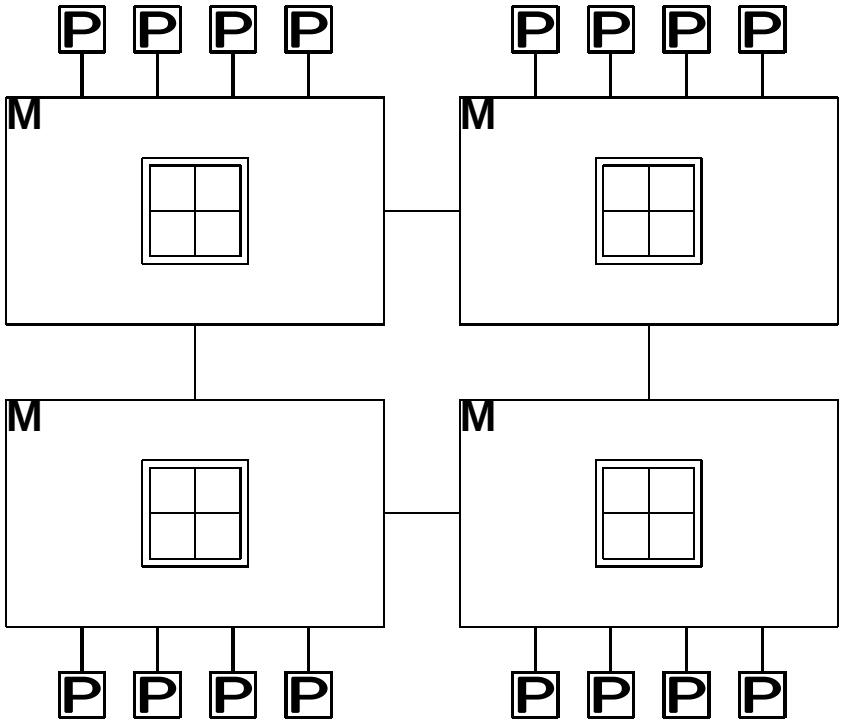
Machine model A machine model describes the hardware and applicable memory semantics: a description of the architecture in terms of **aNode**s and **mNode**s. Execution elements are modeled as made up of *persistent* execution threads (PETs) and *transient* execution threads (TETs).

Decomposition Element A DE is the logical atom of a distributed object, which may or may not map onto a single PET.

Layout A layout describes how DEs are organized along some logical dimensions. We may choose to map these dimensions on to shared and distributed memory.

Uniform interface to the grid index space: the `distGrid`

A **distGrid** describes how to distribute the index space of a component across a DE layout, and most importantly, the *data dependencies* between DEs.

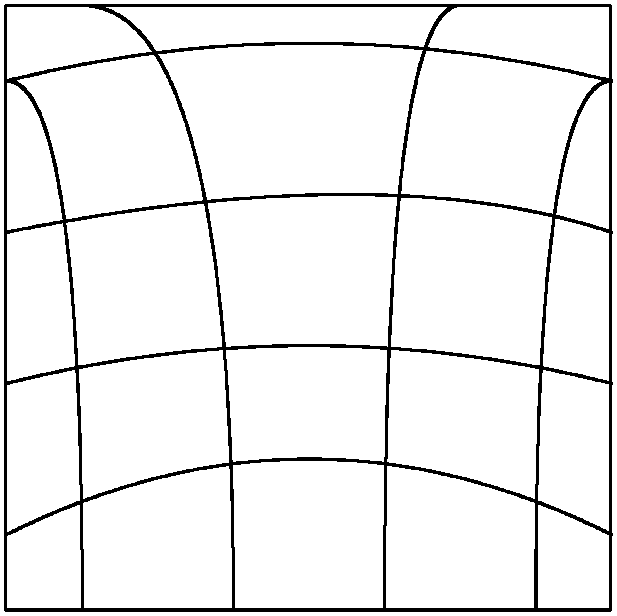


Halo updates and redistributions are typical operations performed using `distGrid`s.

Overlay physical information on the `distGrid`:

`physGrid`

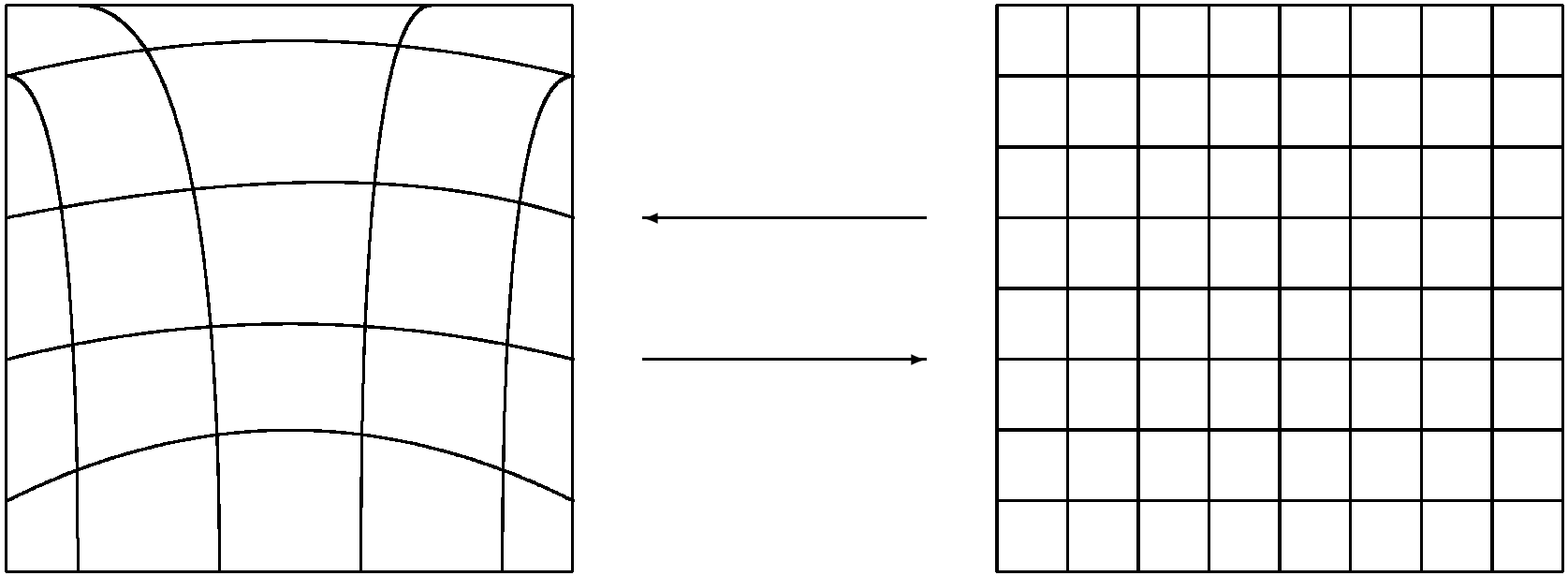
All the model numerics require knowledge of the physical locations, orientations, and interrelations of array indices.



Differential operators (gradient, divergence) are typical operations requiring `physGrid`s.

Interpolate data between grids: `reGrid`

When components on independent `physGrid`s must exchange data, they require a `reGrid`.



On parallel systems, we require all calls to be local; the `reGrid` object contains the optimized **route**s for communication.

Fields and bundles

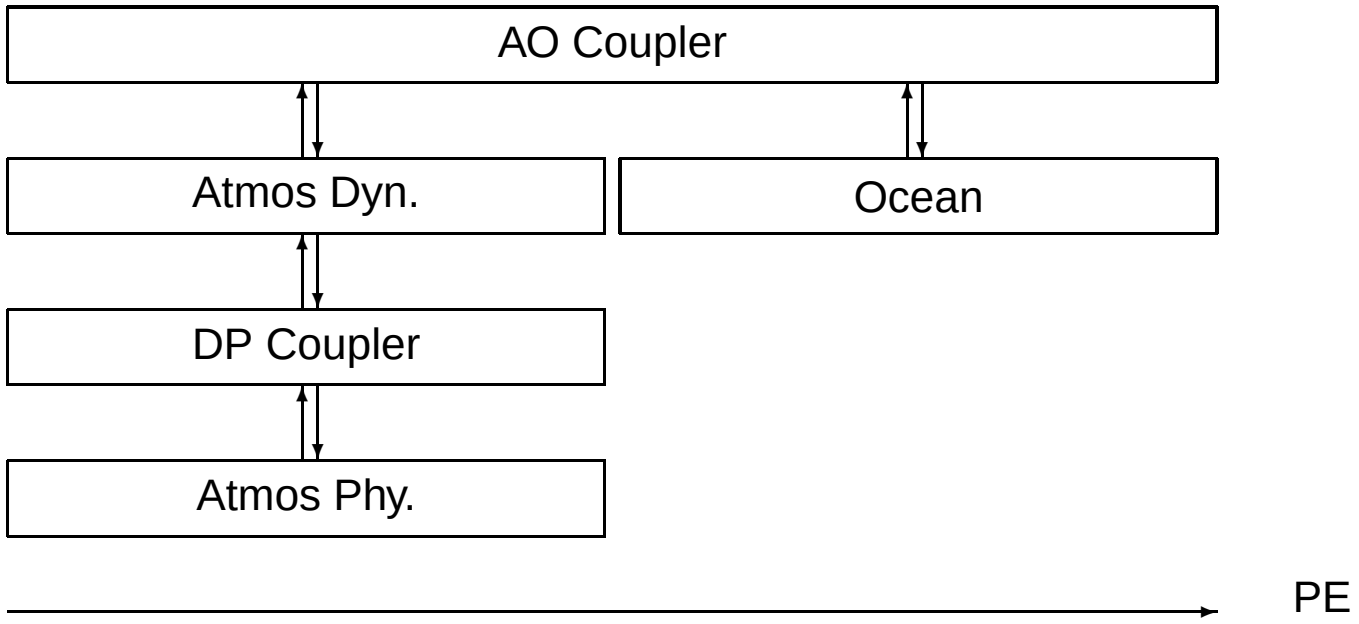
Alongside the grid information, each model variable is present as a distributed array. It is useful, indeed essential, to associate **metadata** that describe the physical content (name, units, range of valid values) of the array. All this information together comprises a model **Field**.

Fields that share a grid may be treated collectively for certain grid operations. A collection of fields sharing a grid is called a **bundle**.

Model composition

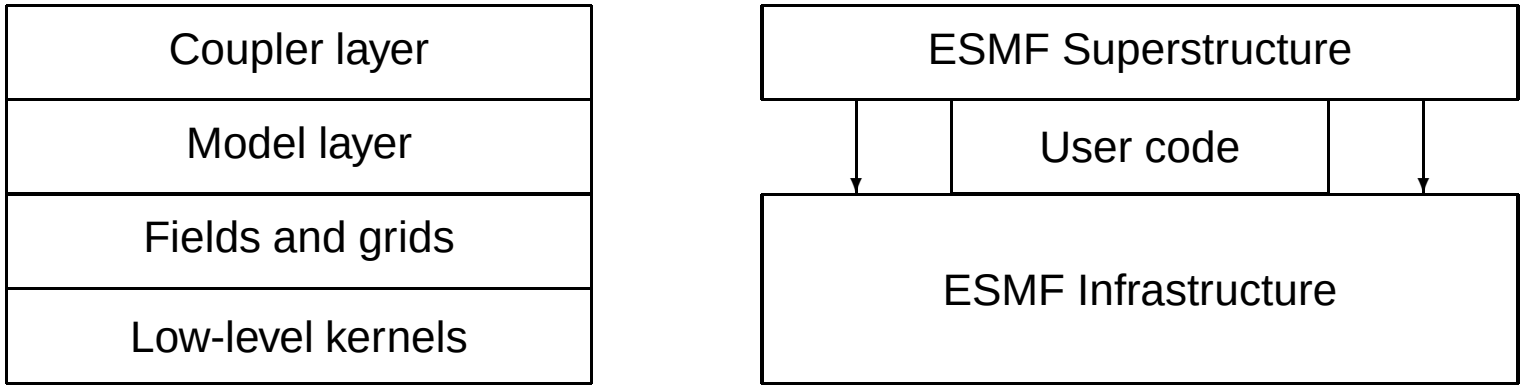
We develop a hierarchy of components and assign them to PETs as appropriate.

A **coupler** mediates between pairs of components needing to exchange data, and matches fields in their **import** and **export States**.



The coupler runs on the union of PETs of its components.

Architecture of an Earth System Modeling Framework: the sandwich

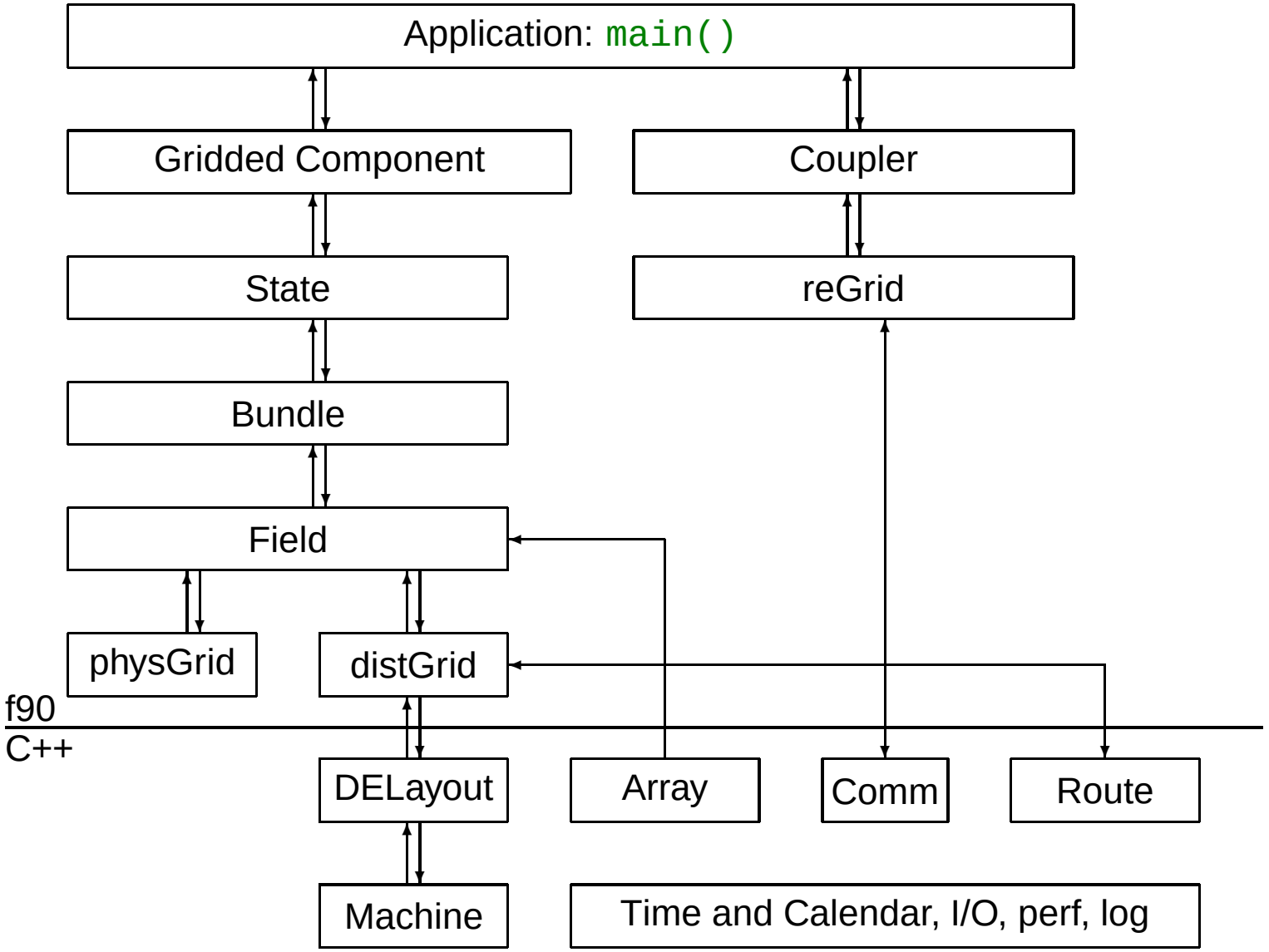


ESMF

The Earth System Modeling Framework is an end-to-end solution for the problem outlined here: supporting distributed development of models with many interacting components, with independent space and time discretization, running on complex modern scalable architectures. A capsule history of ESMF:

- The need to unify and extend current frameworks achieves wide currency (c. 1998).
- NASA offers to underwrite the development of an open community framework (1999).
- A broad cross-section of the community meets and agrees to develop a concerted response to NASA uniting climate models, operational weather models, and data assimilation systems in a common framework (August 1999). Participants include NASA/GMAO, NOAA/GFDL, NOAA/NCEP, NCAR, DOE, and universities with major models.
- Funding began February 2002: $10 M over 3 years.
- First Community Meeting, Washington, May 2002: requirements review.
- Second Community Meeting, Princeton, May 2003: design review.
- Third Community Meeting, Boulder, July 2004: prototype release.

ESMF Class Hierarchy



ESMF features

- ESMF is usable by models written in f90/C++.
- ESMF is usable by models requiring differentiability.
- ESMF is usable by models using shared or distributed memory parallelism semantics.
- ESMF supports serial and concurrent coupling.
- ESMF supports multiple I/O formats (including GRIB/BUFR, netCDF, HDF, native binary).
- ESMF has uniform syntax across platforms.
- ESMF runs on many platforms spanning desktops (laptops, even!) to supercomputers.

Language interoperability

Models increasingly are built to use high-level abstract language features (f90, C++) to facilitate development process across large teams. ESMF is written to be usable by models written in both languages.

Incompatibilities:

- Languages use entirely different memory semantics. ESMF distinguishes between “deep” and “shallow” objects: depending on whether memory is managed by callee or caller.

- High-level data structures do not cross language boundaries. ESMF implements cross-language data structures by implementing an intermediate layer in a low-level language (f77, C) where intrinsic datatypes can be cross-matched. This includes a unique implementation allowing f90 array pointers to be manipulated in either language.

The ESMF **Implementation Report** presents details of its cross-language technology.

Selected web references

http://www.esmf.ucar.edu General website for ESMF: documentation, code, examples, contacts.

http://prism.enes.org General website for PRISM: PRogram for Integrated Earth System Modeling.

http://www.gfdl.noaa.gov/~fms The GFDL Flexible Modeling System. Also links production models and climate model simulation data.

http://gmao.gsfc.nasa.gov/index.php NASA Global Modeling and Assimilation Office: focus on short-term climate variability.

http://mitgcm.org The MIT GCM. Considerable emphasis on data assimilation.

http://www.wrf-model.org The NCAR/NOAA Weather Research and Forecasting model.

http://www.ccsm.ucar.edu The NCAR Community Climate System Model.

http://www.ipcc.ch Intergovernmental Panel on Climate Change. Comprehensive scientific synthesis of current thinking on climate.